REMARKS

Status of the Claims

- Claims 1-8 and 10-15 are pending in the Application.
- Claims 1-8 and 10-15 stand rejected.
- Claim 1 is currently amended by Applicant.

Claim Rejections Pursuant to 35 U.S.C. §112

Claim 1 stands rejected under 35 U.S.C. §112 as being indefinite for use of the term "executable". Applicant amends Claim 1 to remove the term "executable" and to positively recite the actions of the method of Claim 1 which converts an XML intermediate language representation to at least one query which, when executed, instructs the system to query over a plurality of data sources having differing data models. Applicant respectfully requests withdrawal and reconsideration of the 35 U.S.C. §112 rejection of Claim 1.

Claim Rejections Pursuant to 35 U.S.C. §101

Claims 1-8 stand rejected under 35 U.S.C. §101 as being directed to non-statutory subject matter. The Examiner asks to please clarify the method steps within the claim to delete any ambiguity. Applicant amends independent Claim 1 to recite, as noted above, a method of using an API to convert an XML intermediate language representation to at least one query which, when executed, instructs the system to query over a plurality of data sources having differing data models. Applicant respectfully submits that amended Claim 1 is not software per se because the method performs the act of instructing a computer system to query over a plurality of data sources having differing data models. Applicant respectfully requests withdrawal and reconsideration of Claims 1-8 because the claims are directed to a method in a computer system which performs the useful task of querying over multiple data sources.

Claim Rejections Pursuant to 35 U.S.C. §103

Claims 1-2, 5-7, 10-12, and 14-15 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Pat. Ser. No. 6,957,214 to Silberberg et al. (Silberberg) in view of

U.S. Pat. Ser. No. 7,120,645 to Manikutty et al. (Manikutty). Applicant respectfully traverses the rejection.

Silberberg discloses:

> "A system for accessing information from data sources. A user domain translates queries from users and applications for recognition by an aggregation domain, receives responses from the aggregation domain, and translates the responses for recognition by the users and applications. An aggregation domain receives translated queries from the user domain, translates the queries for recognition by a data source domain, receives responses from the data source domain, translates the responses for recognition by the user domain, and transmits the translated responses to the user domain. A data source domain receives the translated queries from the aggregation domain, identifies data sources to receive the queries, translates the queries to the data sources, receives responses from the data sources, translates the responses, and transmits the translated responses to the aggregation domain. A knowledge base the domains use to automatically function includes data models and conceptual terminology translations." (Silberberg, Abstract)

Concerning the knowledge base that includes data models, Silberberg teaches that these data models are produced by "domain engineers" and are entered into the ADINA system as models describing a domain of data. Concerning these data ontologies or models, Silberberg teaches at col. 6:

> "To prepare ADINA for intelligent query processing, domain engineers provide a generic model, *or ontology*, of the overall domain of discourse of the applications and data sources to the ADINA knowledge base. The generic model is independent of individual applications and data sources, but captures the semantics of the overall domain through classes, subclasses, properties, and relationships. The generic model provides the structured common representation of the environment that acts as the glue between the user domain models and the data models." (Silberberg, col. 6, lines 55-64)

Silberberg also teaches in col. 7:

> "User domain engineers provide user domain models to the ADINA knowledge base that describe high-level concept mappings to the generic model. The concept mappings translate between familiar terminology, expressed by property names, the roles that they assume with respect to each other, and constraints, of the user domain to property names, roles, and constraints of the generic model. However, the mappings do not map ontology structures between models. This feature enables systems to be resilient to structural changes of the generic and underlying data source models." (Silberberg, col. 7, lines 19-29)

Thus, Silberberg teaches that domain engineers generate the models or ontology concerning data in domains. The data models of the sources can be mapped to the terminology of the generic ontology model. Silberberg teaches at col. 7:

> "Similarly, data sources register with the ADINA knowledge base by providing the structured data models of their respective data sources as well as high-level, structure-independent concept mappings from the terminology of the data models to the terminology of the generic model. This, too, enables systems to be resilient to data source model changes.
>
> For example, FIG. 4 represents a semi-structured ontology of an XML Student-Faculty document set, which will be described later. It also defines the inter-domain concept mappings between the XML document set and the generic ontology. Similarly, FIG. 5 represents an extended-Entity Relationship (EER) schema of a Grade database as well as its inter-domain concept mappings to the generic ontology." (Silberberg, col. 7 lines 46- 59).

Applicant submits that Silberberg uses domain engineer provided ontologies or models, which may appear as maps or graphs, to describe relationships concerning the data itself in domains. This is distinct from the pending claims which use a graph structure as part of an intermediate language representation of the meaning of an input query. Whereas Silberberg uses an ontology to describe an organization data within a domain, the pending

claims use a graph structure to describe a query. Applicant respectfully submits that there is a difference between an organization of data and a representation of a query.

Thus, Applicant notes that Silberberg fails to teach the Claim 1 and 10 element of: "wherein the XML intermediate language representation is a composite of the plurality of input queries, is an explicit representation of the meaning of the plurality of input queries, and has a graph structure". Applicant also notes that Manikutty also fails to teach this element.

Figure 3 of Manikutty is a flow diagram that illustrates a method of rewriting a database query that has an XML operation. (see col. 4, lines 52-54). Figure 4 of Manikutty is a flow diagram of step 320 of Figure 3. With regard to Step 450 of Figure 4, Manikutty teaches at col. 20 line 61- col. 21, line 17:

"Tree of Canonical XML Generation Functions

> In step 450, canonical XML generation functions are expanded to a normalized tree of canonical functions. In other embodiments, the primitive XML generation operations are expanded to a normalized tree of primitive operations. In a normalized tree, each XEL function has a single non-XML type operand, such as a single scalar operand, or a single abstract data type (ADT) operand, or a single collection type column (typically at a leaf node of the normalized tree), or a set of operands that are of XML type (typically at a parent node of the normalized tree). In some embodiments, a set of operands of XML type might be the result of a tree of canonical XML generation functions. It is ensured that each child in the normalized tree is explicitly tagged. If the child is an ADT that is not tagged, the XEL function is converted to the form XEL (null, "<value expression>" as "<type name>"). Scalar operands are always tagged. For example, FIG. 5 is a block diagram that illustrates a normalized tree 510 of XML generation operations in the sub-query for view dept_xv. In tree 510, each XEL function has a single non-XML type operand, or a set of XML type operands."
> (col. 20, line 61 through col. 21, line 17).

Thus, Applicant notes that Manikutty relies on a tree structure to map in canonical XML generation functions. Applicant also notes that Manikutty is absent any teaching of the use of a graph structure as characteristic of an intermediate language representation as recited in pending Claims 1 and 10.

Applicant notes that Claims 1 and 10 rely on a graph structure for the intermediate language representation. As a reference, the as-filed specification states, in paragraph 0038, that:

> "The XML intermediate language generated as depicted in Figure 2 is a representation of an XML query or view. As such, it may be termed a query intermediate language (QIL) because it is an explicit representation of the meaning of an XML query. The query intermediate language (QIL) may be viewed as a semantic representation common across all XML query and view language compilers. QIL is similar to an ordinary abstract syntax tree (AST) but different in that QIL captures not the syntax of the language but the semantics, or meaning, of a query. Another difference is that QIL is a graph structure and not a tree structure like AST." (present application, para. 0038)

Applicant respectfully submits that the "graph structure", as used in Claims 1 and 10, is specifically not a tree structure because it is so defined in the as-filed specification. Whereas Manikutty explicitly relies on a tree structure to express canonical XML generation functions (see cols 20-21), Claims 1 and 10 explicitly recite the use of a graph structure which is defined in the as-filed specification as distinct from a tree structure.

Silberberg teaches a ontology of data, but not an intermediate language representation of an input query having a graph structure as in Claims 1 and 10. Manikutty teaches an XML generation function expanded to a normalized tree of canonical functions, but does not teach an intermediate language representation of an input query having a graph structure as in independent Claims 1 and 10.

Since both Silberberg and Manikutty, considered separately or combined, fail to teach or suggest an XML intermediate language representation, which is the explicit representation of the meaning of a query and has a graph structure, then the combination of Silberberg and Manikutty cannot render obvious Claims 1 and 10 under 35 U.S.C. §103(a) per MPEP §2143.03 because all claim limitations are not taught by the combination.

Accordingly, Applicant submits Claims 1-8 and 10-15 patentably define over the cited art. Applicant respectfully requests withdrawal of the 35 U.S.C. §103(a) rejection of independent Claims 1 and 10 and their respective dependent claims.

Conclusion

In view of the above remarks and amendments, Applicant respectfully requests withdrawal of the present rejections and reconsideration of the pending claims as they patentably define over the cited art.

Respectfully submitted,

Date: May 9, 2007

/Jerome G. Schaefer/

Jerome G. Schaefer
Registration No. 50,800

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439